



Ambulatory Care Reporting Project

Interface Toolkit (AIT)

January 1998

Department of Veterans Affairs
Veterans Health Administration
Albany CIO Field Office

Table of Contents

Introduction.....	1
Application Programmer Interfaces.....	3
56 - SDOE Get Diagnoses	3
58 - SDOE Get Providers	5
61 - SDOE Get Procedures	7
63 - SDOE Assigned a Provider	9
64 - SDOE Assigned a Diagnosis	10
65 - SDOE Assigned a Procedure	11
69 - SDOE Find Provider	12
70 - SDOE Find Diagnosis	13
71 - SDOE Find Procedure	14
72 - SDOE Find First Standalone	15
73 - SDOE Get Primary Diagnosis	16
74 - SDOE Find First Encounter	17
75 - SDOE Find Last Standalone	18
76 - SDOE Get General Data	19
78 - SDOE Parse General Data	20
79 - SDQ Open	21
80 - SDQ Close	22
81 - SDQ Patient	23
82 - SDQ Date Range	24
83 - SDQ Filter	25
84 - SDQ Visit	27
85 - SDQ Index Name	28
86 - SDQ EOF	29
87 - SDQ BOF	30
88 - SDQ Active Status	31
89 - SDQ Count	33
90 - SDQ First	34
91 - SDQ Last	35
92 - SDQ Next	36
93 - SDQ Prior	37
94 - SDQ Refresh.....	38
95 - SDQ Get Current Entry ID	39
98 - SDOE Get Zero Node	40
99 - SDQ Scan	41
100 - SDQ Scan Callback.....	42
101 - SDQ Error Check	43

Table of Contents

Remote Procedure Calls	45
SDOE Assigned a Diagnosis	45
SDOE Assigned a Procedure	46
SDOE Assigned a Provider	47
SDOE Find Diagnosis	48
SDOE Find First Encounter	49
SDOE Find First Standalone	50
SDOE Find Last Standalone	51
SDOE Find Procedure	52
SDOE Find Provider	53
SDOE Get Diagnoses	54
SDOE Get General Data.....	55
SDOE Get Primary Diagnosis	56
SDOE Get Procedures.....	57
SDOE Get Providers	58
SDOE Get Zero Node	59
SDOE List Encounters for Dates	60
SDOE List Encounters for PAT	61
SDOE List Encounters for Visit	62
SDOE Parse General Data	63
Error Processing	65
Structure	65
Error Arrays	65
AIT Error Processing Tools.....	66
Checking for Errors	66
Debugging	67

Appendix - Argument Definitions.....	69
Action.....	69
Begin Date/Time	69
CPT IEN	69
Diagnosis IEN	70
Direction	70
Encounter Data	70
Encounter IEN	71
Encounter Parse Format.....	71
Encounter Parsed Data.....	72
Encounter Query Active Status.....	72
Encounter Query Filter	73
Encounter Query Handle.....	73
Encounter Query Index	74
End Date/Time	74
Error Array.....	75
List of V CPT Entries	75
List of V POV Entries	76
List of V PROVIDER Entries	76
Patient ID	76
Practitioner ID	77
Scan Callback Logic	77
Search Flags.....	78
Visit IEN.....	78

Table of Contents

Introduction

The ACRP Interface Toolkit (AIT) is a set of programmer tools that provides access to outpatient encounter data. This initial version contains Application Programmer Interfaces (APIs) and Remote Procedure Calls (RPCs) that provide access to procedure, diagnosis, provider, and general data related to an encounter. It is hoped that in a future version of the AIT, Delphi objects, components, and DLLs will be provided as well.

This AIT provides Class I packages, Class III software, and other local code with one highly structured interface to the encounter data.

Introduction

Application Programmer Interfaces

ID: 56

Name: SDOE GET DIAGNOSES

Declaration: GETDX^SDOE(encounter, dx_list [,errors])

Description: This procedure returns a subscripted array of diagnoses for an encounter.

The array subscripts are arbitrary integer numbers.

Each array entry equals the zeroth node of a record in the V POV file.

The top level, non-subscripted value of the array variable is the count of the number of V POV entries in the array.

NOTE 1: For encounters before 10/1/96, only scheduling data in the OUTPATIENT DIAGNOSIS (#409.43) file may exist. It will only exist if the site required diagnoses as part of the check-out process.

This API will attempt to find this “old” data, reformat the data to meet the V POV structure, and return the list of diagnoses as stated in the above description. (Only the diagnosis code internal entry number is available for “old” encounters).

NOTE 2: Currently, each array entry corresponds to an entry in the V POV file. In the future, it is possible that **VISTA** will not use V POV as the source of diagnostic data. However, if another source is used, this API will reformat that source and present the data in the V POV format so as not to affect the use of the API by calling applications.

Arguments:

encounter	Encounter IEN
dx_list	List of V POV Entries
errors	Error Array [optional]

Return Values: Not applicable for procedures

Related RPC: SDOE GET DIAGNOSES

Error Codes: 4,096,800.001 Invalid Encounter ID

Example:

```
N DXLIST,I  
>>> D GETDX^SDOE(4592,"DXLIST")  
S I=0  
F S I=$O(DXLIST(I)) Q:'I W !,I,:",DXLIST(I)  
W !,"Count: ",+$G(DXLIST)  
--> 370: 97^101^459^192^^^^^^^P  
Count: 1
```

ID: 58**Name:** SDOE GET PROVIDERS**Declaration:** GETPRV^SDOE(encounter, provider_list [,errors])**Description:** This procedure returns a subscripted array of providers for an encounter.

The array subscripts are arbitrary integer numbers.

Each array entry equals the zeroth node of a record in the V PROVIDER file.

The top level, non-subscripted value of the array variable is the count of the number of V PROVIDER entries in the array.

NOTE 1: For encounters before 10/1/96, only scheduling data in the OUTPATIENT PROVIDER (#409.44) file may exist. It will only exist if the site required provider as part of the check-out process.

This API will attempt to find this “old” data, reformat the data to meet the V PROVIDER structure, and return the list of providers as stated in the above description. (Only the provider internal entry number is available for “old” encounters).

NOTE 2: Currently, each array entry corresponds to an entry in the V PROVIDER file. In the future, it is possible that **VISTA** will not use V PROVIDER as the source of provider data. However, if another source is used, this API will reformat that source and present the data in the V PROVIDER format so as not to affect the use of the API by calling applications.

Arguments:

encounter	Encounter IEN
provider_list	List of V PROVIDER Entries
errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** SDOE GET PROVIDERS

Application Programmer Interfaces

Error Codes: 4,096,800.001 Invalid Encounter ID

Example:

```
N PRVLIST,I  
>>> D GETPRV^SDOE(4592,"PRVLIST")  
S I=0  
F S I=$O(PRVLIST(IEN)) Q:'I W !,I,:",PRVLIST(I)  
W !,"Count: ",+$G(PRVLIST)  
  
--> 284: 11344^706^407^P^^11  
Count: 1
```

ID: 61**Name:** SDOE GET PROCEDURES**Declaration:** GETCPT^SDOE(encounter, cpt_list [,errors])**Description:** This procedure returns a subscripted array of CPTs for an encounter.

The array subscripts are arbitrary integer numbers.

Each array entry equals the zeroth node of a record in the V CPT file.

The top level, non-subscripted value of the array variable is the count of the number of V CPT entries in the array.

NOTE 1: For encounters before 10/1/96, only scheduling data in the SCEDULING VISITS (#409.5) file may exist. It will only exist if the site required procedures as part of the check-out process.

This API will attempt to find this “old” data, reformat the data to meet the V CPT structure, and return the list of procedures as stated in the above description. (Only the CPT code internal entry number and count are available for “old” encounters).

NOTE 2: Currently, each array entry corresponds to an entry in the V CPT file. In the future, it is possible that **VISTA** will not use V CPT as the source of procedure data. However, if another source is used, this API will reformat that source and present the data in the V CPT format so as not to affect the use of the API by calling applications.

Arguments:	encounter	Encounter IEN
	cpt_list	List of V CPT Entries
	errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** SDOE GET PROCEDURES

Error Codes: 4,096,800.001 Invalid Encounter ID

Example:

```
N CPTLIST,I  
>>> D GETCPT^SDOE(4592,"CPTLIST")  
S I=0  
F S I=$O(CPTLIST(I)) Q:'I W !,I,: ",CPTLIST(I)  
W !,"Count: ",+$G(CPTLIST)  
  
--> 352: 10060^706^407^69^^^^^^^^^^^^^1  
353: 10061^706^407^69^^^^^^^^^^^^^1  
Count: 2
```

ID: 63

Name: SDOE ASSIGNED A PROVIDER

Declaration: \$\$PRV^SDOE(encounter [,errors])

Description: This function returns a boolean indicator on whether at least one provider has been associated with an encounter.

Arguments:

encounter	Encounter IEN
errors	Error Array [optional]

Return Values:

- 1 - Yes, at least one provider is associated with encounter
- 0 - No, no providers are associated with encounter

Related RPC: SDOE ASSIGNED A PROVIDER

Error Codes: 4,096,800.001 Invalid Encounter ID

Example:

```
>>> W $$PRV^SDOE( 4592 )
--> 1
```

ID: 64

Name: SDOE ASSIGNED A DIAGNOSIS

Declaration: \$\$DX^SDOE(encounter [,errors])

Description: This function returns a boolean indicator on whether at least one diagnoses has been associated with an encounter.

Arguments: encounter Encounter IEN
 errors Error Array [optional]

Return Values: 1 - Yes, at least one diagnosis is associated with encounter

0 - No, no diagnoses are associated with encounter

Related RPC: SDOE ASSIGNED A DIAGNOSIS

Error Codes: 4,096,800.001 Invalid Encounter ID

Example: >>> W \$\$DX^SDOE(4592)

--> 1

ID: 65

Name: SDOE ASSIGNED A PROCEDURE

Declaration: \$\$CPT^SDOE(encounter [,errors])

Description: This function returns a boolean indicator on whether at least one procedure has been associated with an encounter.

Arguments: encounter Encounter IEN
 errors Error Array [optional]

Return Values: 1 - Yes, at least one procedure is associated with encounter

0 - No, no procedures are associated with encounter

Related RPC: SDOE ASSIGNED PROCEDURE

Error Codes: 4,096,800.001 Invalid Encounter ID

Example: >>> W \$\$CPT^SDOE(4592)

--> 1

ID:	69						
Name:	SDOE FIND PROVIDER						
Declaration:	<code>\$\$FINDPRV^SDOE(encounter, provider [,errors])</code>						
Description:	This function returns a boolean indicator on whether a specific provider is associated with an encounter.						
Arguments:	<table><tr><td>encounter</td><td>Encounter IEN</td></tr><tr><td>provider</td><td>Practitioner ID</td></tr><tr><td>errors</td><td>Error Array [optional]</td></tr></table>	encounter	Encounter IEN	provider	Practitioner ID	errors	Error Array [optional]
encounter	Encounter IEN						
provider	Practitioner ID						
errors	Error Array [optional]						
Return Values:	1 - Yes, specific provider is associated with encounter 0 - No, provider is not associated with encounter						
Related RPC:	SDOE FIND PROVIDER						
Error Codes:	4,096,800.001 Invalid Encounter ID 4,096,800.003 Invalid Provider ID						
Example:	<pre>>>> W \$\$FINDPRV^SDOE(4592,990) --> 1</pre>						

ID:	70						
Name:	SDOE FIND DIAGNOSIS						
Declaration:	<code>\$\$FINDDX^SDOE(encounter, diagnosis [,errors])</code>						
Description:	This function returns a boolean indicator on whether a specific diagnosis is associated with an encounter.						
Arguments:	<table><tr><td>encounter</td><td>Encounter IEN</td></tr><tr><td>diagnosis</td><td>Diagnosis IEN</td></tr><tr><td>errors</td><td>Error Array [optional]</td></tr></table>	encounter	Encounter IEN	diagnosis	Diagnosis IEN	errors	Error Array [optional]
encounter	Encounter IEN						
diagnosis	Diagnosis IEN						
errors	Error Array [optional]						
Return Values:	1 - Yes, specific diagnosis is associated with encounter 0 - No, diagnosis is not associated with encounter						
Related RPC:	SDOE FIND DIAGNOSIS						
Error Codes:	4,096,800.001 Invalid Encounter ID 4,096,800.004 Invalid Diagnosis ID						
Example:	<pre>>>> W \$\$FINDDX^SDOE(4592,35) --> 1</pre>						

ID:	71						
Name:	SDOE FIND PROCEDURE						
Declaration:	<code>\$\$FINDCPT^SDOE(encounter, cpt [,errors])</code>						
Description:	This function returns a boolean indicator on whether a specific procedure is associated with an encounter.						
Arguments:	<table><tr><td>encounter</td><td>Encounter IEN</td></tr><tr><td>cpt</td><td>CPT IEN</td></tr><tr><td>errors</td><td>Error Array [optional]</td></tr></table>	encounter	Encounter IEN	cpt	CPT IEN	errors	Error Array [optional]
encounter	Encounter IEN						
cpt	CPT IEN						
errors	Error Array [optional]						
Return Values:	1 - Yes, specific procedure is associated with encounter 0 - No, procedure is not associated with encounter						
Related RPC:	SDOE FIND PROCEDURE						
Error Codes:	4,096,800.001 Invalid Encounter ID 4,096,800.005 Invalid CPT ID						
Example:	<pre>>>> W \$\$FINDCPT^SDOE(4592,10061) --> 1</pre>						

ID:	72										
Name:	SDOE FIND FIRST STANDALONE										
Declaration:	<code>\$\$EXAE^SDOE(dfn, begin_date, end_date [,flags][,errors])</code>										
Description:	This function returns the internal entry number of an OUTPATIENT ENCOUNTER file (#409.68) entry for the first standalone add/edit for a patient in a specified date range.										
	Use same date for begin and end dates for specific (single) date check.										
	Standalone encounter is an encounter with no parent and the originating process is "Stop Code Addition".										
Arguments:	<table border="0"> <tr> <td>dfn</td><td>Patient ID</td></tr> <tr> <td>begin_date</td><td>Begin Date/Time</td></tr> <tr> <td>end_date</td><td>End Date/Time</td></tr> <tr> <td>flags</td><td>Search Flags</td></tr> <tr> <td>errors</td><td>Error Array [optional]</td></tr> </table>	dfn	Patient ID	begin_date	Begin Date/Time	end_date	End Date/Time	flags	Search Flags	errors	Error Array [optional]
dfn	Patient ID										
begin_date	Begin Date/Time										
end_date	End Date/Time										
flags	Search Flags										
errors	Error Array [optional]										
Return Values:	<table border="0"> <tr> <td><pointer></td><td>Outpatient Encounter IEN for first standalone encounter found in date range</td></tr> <tr> <td><null></td><td>if no encounter exists</td></tr> </table>	<pointer>	Outpatient Encounter IEN for first standalone encounter found in date range	<null>	if no encounter exists						
<pointer>	Outpatient Encounter IEN for first standalone encounter found in date range										
<null>	if no encounter exists										
Related RPC:	SDOE FIND FIRST STANDALONE										
Error Codes:	<table border="0"> <tr> <td>4,096,800.002</td><td>Invalid Patient ID</td></tr> <tr> <td>4,096,800.022</td><td>Invalid Date Range</td></tr> </table>	4,096,800.002	Invalid Patient ID	4,096,800.022	Invalid Date Range						
4,096,800.002	Invalid Patient ID										
4,096,800.022	Invalid Date Range										
Example:	<pre>>>> W \$\$EXAE^SDOE(101,2970501,2970601,"C")</pre> <pre>--> 4501</pre>										

ID: 73

Name: SDOE GET PRIMARY DIAGNOSIS

Declaration: \$\$GETPDX^SDOE(encounter [,errors])

Description: This function returns the internal entry number of the primary diagnosis code (^ICD9) for an encounter.

NOTE: For encounters before 10/1/96, this function will always return 0. This primary diagnosis was not retrieved nor stored by the system for these “old” encounters.

Arguments: encounter Encounter IEN
 errors Error Array [optional]

Return Values: <pointer> ien to ^ICD9 for primary dx
 0 no primary dx found for encounter

Related RPC: SDOE GET PRIMARY DIAGNOSIS

Error Codes: 4,096,800.001 Invalid Encounter ID
 4,096,800.025 Duplicate Primary Diagnosis

Example: >>> W \$\$GETPDX^SDOE(4592)

--> 35

ID:	74										
Name:	SDOE FIND FIRST ENCOUNTER										
Declaration:	<code>\$\$EXOE^SDOE(dfn, begin_date, end_date [,flags][,errors])</code>										
Description:	This function returns the internal entry number of an OUTPATIENT ENCOUNTER file (#409.68) entry for the first encounter for a patient in a specified date range.										
Arguments:	<table><tr><td>dfn</td><td>Patient ID</td></tr><tr><td>begin_date</td><td>Begin Date/Time</td></tr><tr><td>end_date</td><td>End Date/Time</td></tr><tr><td>flags</td><td>Search Flags</td></tr><tr><td>errors</td><td>Error Array [optional]</td></tr></table>	dfn	Patient ID	begin_date	Begin Date/Time	end_date	End Date/Time	flags	Search Flags	errors	Error Array [optional]
dfn	Patient ID										
begin_date	Begin Date/Time										
end_date	End Date/Time										
flags	Search Flags										
errors	Error Array [optional]										
Return Values:	<table><tr><td><pointer></td><td>Outpatient Encounter ID for first encounter found in date range</td></tr><tr><td><null></td><td>if no encounter exists</td></tr></table>	<pointer>	Outpatient Encounter ID for first encounter found in date range	<null>	if no encounter exists						
<pointer>	Outpatient Encounter ID for first encounter found in date range										
<null>	if no encounter exists										
Related RPC:	SDOE FIND FIRST ENCOUNTER										
Error Codes:	<table><tr><td>4,096,800.002</td><td>Invalid Patient ID</td></tr><tr><td>4,096,800.022</td><td>Invalid Date Range</td></tr></table>	4,096,800.002	Invalid Patient ID	4,096,800.022	Invalid Date Range						
4,096,800.002	Invalid Patient ID										
4,096,800.022	Invalid Date Range										
Example:	<pre>>>> W \$\$EXOE^SDOE(101,2970501,2970601,"C") --> 4504</pre>										

ID:	75								
Name:	SDOE FIND LAST STANDALONE								
Declaration:	<code>\$\$GETLAST^SDOE(dfn, begin_date [,flags][,errors])</code>								
Description:	This function returns the internal entry number of an OUTPATIENT ENCOUNTER file (#409.68) entry for the last standalone add/edit for a patient. All encounters from the specified begin date to the current date are used to find this last standalone.								
	Standalone encounter is an encounter with no parent and the originating process is "Stop Code Addition".								
Arguments:	<table><tr><td>dfn</td><td>Patient ID</td></tr><tr><td>begin_date</td><td>Begin Date/Time</td></tr><tr><td>flags</td><td>Search Flags</td></tr><tr><td>errors</td><td>Error Array [optional]</td></tr></table>	dfn	Patient ID	begin_date	Begin Date/Time	flags	Search Flags	errors	Error Array [optional]
dfn	Patient ID								
begin_date	Begin Date/Time								
flags	Search Flags								
errors	Error Array [optional]								
Return Values:	<table><tr><td><pointer></td><td>Outpatient Encounter ID for last standalone encounter found after date</td></tr><tr><td><null></td><td>if no encounter exists</td></tr></table>	<pointer>	Outpatient Encounter ID for last standalone encounter found after date	<null>	if no encounter exists				
<pointer>	Outpatient Encounter ID for last standalone encounter found after date								
<null>	if no encounter exists								
Related RPC:	SDOE FIND LAST STANDALONE								
Error Codes:	<table><tr><td>4,096,800.002</td><td>Invalid Patient ID</td></tr><tr><td>4,096,800.022</td><td>Invalid Date Range</td></tr></table>	4,096,800.002	Invalid Patient ID	4,096,800.022	Invalid Date Range				
4,096,800.002	Invalid Patient ID								
4,096,800.022	Invalid Date Range								
Example:	<pre>>>> W \$\$GETLAST^SDOE(101,2970414,"C") --> 4559</pre>								

ID: 76**Name:** SDOE GET GENERAL DATA**Declaration:** GETGEN^SDOE(encounter, encounter_data [,errors])**Description:** This procedure returns the zeroth and other nodes of an outpatient encounter entry.

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the zeroth node are returned.

For detail information regarding the fields, see the data dictionary for the OUTPATIENT ENCOUNTER file (#409.68).

Arguments:

encounter	Encounter IEN
encounter_data	Encounter Data
errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** SDOE GET GENERAL DATA**Error Codes:** 4,096,800.001 Invalid Encounter ID

Example:

```

N DATA,NODE,XXERR
>>> D GETGEN^SDOE( 4592, "DATA", "XXERR" )
S NODE="" F S NODE=$O(DATA(NODE)) Q:NODE="" D
. W !,"Node: ",NODE," = ",DATA(SDNODE)

--> Node: 0 = 2970602.08^706^144^62^407^^
2970805.1107^^^9^1^2^10

```

ID:	78								
Name:	SDOE PARSE GENERAL DATA								
Declaration:	PARSE^SDOE(.encounter_data, format, parsed_data [,errors])								
Description:	This procedure will parse the data returned by the “SDOE GET GENERAL DATA” API into individual field nodes.								
	The parser will either use internal or external values for the field node values. The developer specifies internal/external via a parameter.								
Arguments:	<table><tr><td>encounter_data</td><td>Encounter Data</td></tr><tr><td>format</td><td>Encounter Parse Format</td></tr><tr><td>parsed_data</td><td>Encounter Parsed Data</td></tr><tr><td>errors</td><td>Error Array [optional]</td></tr></table>	encounter_data	Encounter Data	format	Encounter Parse Format	parsed_data	Encounter Parsed Data	errors	Error Array [optional]
encounter_data	Encounter Data								
format	Encounter Parse Format								
parsed_data	Encounter Parsed Data								
errors	Error Array [optional]								
Return Values:	Not applicable for procedures								
Related RPC:	SDOE PARSE GENERAL DATA								
Error Codes:	4,096,800.023 Invalid Parse Format 4,096,800.024 No Data to Parse								
Example:	<pre>N DATA,NODE,PARSED D GETGEN^SDOE(4592,"DATA") >>> D PARSE^ SDOE(.DATA,"EXTERNAL","PARSE") S NODE="" F S NODE=\$O(PARSE(NODE)) Q:NODE="" D . W !,"Node: " NODE," ==> ",PARSE(NODE) --> Node: .01 ==> Jun 02, 1997@08:00 Node: .02 ==> DAVIS,SUE Node: .03 ==> DERMATOLOGY Node: .04 ==> DERMATOL OGY Node: .05 ==> Jun 02, 1997@08:00 Node: .06 ==> Node: .07 ==> Aug 05, 1997@11:07 Node: .08 ==> APPOINTMENT Node: .1 ==> REGULAR Node: .11 ==> TROY Node: .12 ==> CHECKED OUT Node: .13 ==> NSC</pre>								

ID: 79

Name: SDQ OPEN

Declaration: OPEN^SDQ(.query [,errors])

Description: This method is used by a developer to initialize a Query Object and obtain a query handle.

Arguments:

query	Encounter Query Handle
errors	Error Array [optional]

Return Values: Not applicable for procedures

Related RPC: Not applicable

Error Codes: 4,096,800.109 Invalid Query Property

Example:

```

EN  N QUERY
>>> D OPEN^SDQ( .QUERY )
      IF '$$ERRCHK^SDQUT()' D INDEX^SDQ( .QUERY, "PATIENT/DATE", "SET" )
      IF '$$ERRCHK^SDQUT()' D PAT^SDQ( .QUERY, 101, "SET" )
      IF '$$ERRCHK^SDQUT()' D DATE^SDQ( .QUERY, 2970501, 2970601, "SET" )
      IF '$$ERRCHK^SDQUT()' D SCANCB^SDQ( .QUERY, "D CB^XXSCAN(Y,Y0,
      .SDSTOP)", "SET" )
      IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ( .QUERY, "TRUE", "SET" )
      IF '$$ERRCHK^SDQUT()' D SCAN ^SDQ( .QUERY, "FORWARD" )
      D CLOSE^SDQ( .QUERY )
      Q
;
CB(SDOE,SDOE0,SDSTOP)      ; -- callback logic
W !,SDOE,">>>",SDOE0
Q

```

ID: 80

Name: SDQ CLOSE

Declaration: CLOSE^SDQ(.query [,errors])

Description: This method is used by a developer to close a Query Object.

NOTE: The Encounter Query handle is set to null as a result of a successful call to SDQ CLOSE.

Arguments: query Encounter Query Handle
 errors Error Array [optional]

Return Values: Not applicable for procedures

Related RPC: Not applicable

Error Codes: 4,096,800.101 Invalid Query Object Handle

Example: EN N QUERY
D OPEN^SDQ(.QUERY)
IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET")
IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2970501, 2970601, "SET")
IF '\$\$ERRCHK^SDQUT()' D SCANCB^SDQ(.QUERY, "D CB^XXSCAN(Y,Y0,
.SDSTOP)", "SET")
IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
IF '\$\$ERRCHK^SDQUT()' D SCAN^SDQ(.QUERY, "FORWARD")
>>> D CLOSE^SDQ(.QUERY)
Q
;
CB(SDOE, SDOE0, SDSTOP) ; -- callback logic
W !, SDOE, " >>>", SDOE0
Q

ID: 81**Name:** SDQ PATIENT**Declaration:** PAT^SDQ(query, .dfn, action [,errors])**Description:** This method is used to set and retrieve the Patient property of a Query Object.

In order to activate a query, this property must be set if the Index Name property is either PATIENT or PATIENT/DATE.

Arguments:	query	Encounter Query Handle
	dfn	Patient ID
	action	Action
	errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** Not applicable

Error Codes:	4,096,800.002 Invalid Patient ID
	4,096,800.101 Invalid Query Object Handle
	4,096,800.106 Active Query
	4,096,800.108 Invalid Property Action

Example:

```

EN N QUERY
D OPEN^SDQ(.QUERY)
IF '$$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
>>> IF '$$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET")
IF '$$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2970501, 2970601, "SET")
IF '$$ERRCHK^SDQUT()' D SCANCEB^SDQ(.QUERY, "D CALLBACK^XXSCAN(Y,
    Y0, .SDSTOP)", "SET")
IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
IF '$$ERRCHK^SDQUT()' D SCAN^SDQ(.QUERY, "FORWARD")
D CLOSE^SDQ(.QUERY)
Q
;
CB(SDOE, SDOE0, SDSTOP)      ; - - callback logic
W !, SDOE, " >>>", SDOE0
Q

```

ID:	82										
Name:	SDQ DATE RANGE										
Declaration:	DATE^SDQ(query, .begin_date, .end_date, action [,errors])										
Description:	This method is used to set and retrieve the Date Range property of a Query Object.										
<p>In order to activate a query, this property must be set if the Index Name property is either DATE/TIME or PATIENT/DATE.</p> <p>NOTE: During a “set” action, a developer can pass in a “Begin Date/Time” of zero. This will be converted to January 1, 1990 (VA FileMan internal format of 2900101).</p>											
Arguments:	<table border="0"> <tr> <td>query</td><td>Encounter Query Handle</td></tr> <tr> <td>begin_date</td><td>Begin Date/Time</td></tr> <tr> <td>end_date</td><td>End Date/Time</td></tr> <tr> <td>action</td><td>Action</td></tr> <tr> <td>errors</td><td>Error Array [optional]</td></tr> </table>	query	Encounter Query Handle	begin_date	Begin Date/Time	end_date	End Date/Time	action	Action	errors	Error Array [optional]
query	Encounter Query Handle										
begin_date	Begin Date/Time										
end_date	End Date/Time										
action	Action										
errors	Error Array [optional]										
Return Values:	Not applicable for procedures										
Related RPC:	Not applicable										
Error Codes:	<table border="0"> <tr> <td>4,096,800.022</td><td>Invalid Date Range</td></tr> <tr> <td>4,096,800.101</td><td>Invalid Query Object Handle</td></tr> <tr> <td>4,096,800.106</td><td>Active Query</td></tr> <tr> <td>4,096,800.108</td><td>Invalid Property Action</td></tr> </table>	4,096,800.022	Invalid Date Range	4,096,800.101	Invalid Query Object Handle	4,096,800.106	Active Query	4,096,800.108	Invalid Property Action		
4,096,800.022	Invalid Date Range										
4,096,800.101	Invalid Query Object Handle										
4,096,800.106	Active Query										
4,096,800.108	Invalid Property Action										

Example:

```

EN  N QUERY
D OPEN^SDQ( .QUERY)
IF '$$ERRCHK^SDQUT() D INDEX^SDQ( .QUERY, "PATIENT    /DATE", "SET")
IF '$$ERRCHK^SDQUT() D PAT^SDQ( .QUERY,101, "SET")
>>> IF '$$ERRCHK^SDQUT() D DATE^SDQ( .QUERY,2970501,2970601, "SET")
IF '$$ERRCHK^SDQUT() D SCANCB^SDQ( .QUERY, "D CB^XXSCAN(Y,Y0,
.SDSTOP)", "SET")
IF '$$ERRCHK^SDQUT() D ACTI VE^SDQ( .QUERY, "TRUE", "SET")
IF '$$ERRCHK^SDQUT() D SCAN^SDQ( .QUERY, "FORWARD")
D CLOSE^SDQ( .QUERY)
Q
;
CB(SDOE,SDOE0,SDSTOP)      ; -- callback logic
W !,SDOE," >>>",SDOE0
Q

```

ID: 83**Name:** SDQ FILTER**Declaration:** FILTER^SDQ(query, .filter, action [,errors])**Description:** This method is used to set and retrieve the Filter property of a Query Object.

The following local variables are available at run-time:

Y	current encounter entry number
Y0	zeroth node of current encounter entry (only supported fields)

NOTE: If the developer plans to use the SDQ SCAN method of the query object, it is more efficient to place any “filter-type” logic in the Callback property logic. It is still valid to set the Filter property and call the SDQ SCAN method. However, the method will execute faster if the Callback property logic contains the filter logic.

Arguments:	query	Encounter Query Handle
	filter	Encounter Query Filter
	action	Action
	errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** Not applicable

Error Codes:	4,096,800.101	Invalid Query Object Handle
	4,096,800.104	Invalid Filter
	4,096,800.106	Active Query
	4,096,800.108	Invalid Property Action

Application Programmer Interfaces

Example:

```
EN N QUERY
D OPEN^SDQ(.Q UERY)
IF '$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")
IF '$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")
IF '$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")
IF '$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y 0,
.SDSTOP)","SET")
>>> IF '$$ERRCHK^SDQUT() D FILTER^SDQ(.QUERY,"IF $P(Y0,U,8)=1",
"SET")
IF '$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")
IF '$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")
D CLOSE^SDQ(.QUERY)
Q
;
CB(SDOE,SDOE0,SDSTOP)      ; -- callback logic
W !,SDOE,">>>",SDOE0
Q
```

ID: 84**Name:** SDQ VISIT**Declaration:** VISIT^SDQ(query, .visit, action [,errors])**Description:** This method is used to set and retrieve the Visit property of a Query Object.

In order to activate a query, this property must be set if the Index Name property is set to VISIT.

Arguments:	query	Encounter Query Handle
	visit	Visit IEN
	action	Action
	errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** Not applicable

Error Codes:	1,509,000.001	Invalid Visit IEN
	4,096,800.101	Invalid Query Object Handle
	4,096,800.106	Active Query
	4,096,800.108	Invalid Property Action

Example:

```

EN N QUERY
D OPEN^SDQ(.QUERY)
IF '$$E RRCHK^SDQUT() D INDEX^SDQ(.QUERY,"VISIT","SET")
>>> IF '$$ERRCHK^SDQUT() D VISIT^SDQ(.QUERY,875833,"SET")
IF '$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,
    .SDSTOP)","SET")
IF '$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")
IF '$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")
D CLOSE^SDQ(.QUERY)
Q
;
CB(SDOE,SDOE0,SDSTOP)      ; -- callback logic
W !,SDOE,">>>",SDOE0
Q

```

ID:	85								
Name:	SDQ INDEX NAME								
Declaration:	INDEX^SDQ(query, .index, action [,errors])								
Description:	This method is used to set and retrieve the Index Name property of a Query Object.								
Valid Values									
PATIENT									
PATIENT/DATE									
DATE/TIME									
VISIT									
Arguments:	<table border="0"> <tr> <td>query</td><td>Encounter Query Handle</td></tr> <tr> <td>index</td><td>Encounter Query Index</td></tr> <tr> <td>action</td><td>Action</td></tr> <tr> <td>errors</td><td>Error Array [optional]</td></tr> </table>	query	Encounter Query Handle	index	Encounter Query Index	action	Action	errors	Error Array [optional]
query	Encounter Query Handle								
index	Encounter Query Index								
action	Action								
errors	Error Array [optional]								
Return Values:	Not applicable for procedures								
Related RPC:	Not applicable								
Error Codes:	<table border="0"> <tr> <td>4,096,800.101</td><td>Invalid Query Object Handle</td></tr> <tr> <td>4,096,800.105</td><td>Invalid Index</td></tr> <tr> <td>4,096,800.106</td><td>Active Query</td></tr> <tr> <td>4,096,800.108</td><td>Invalid Property Action</td></tr> </table>	4,096,800.101	Invalid Query Object Handle	4,096,800.105	Invalid Index	4,096,800.106	Active Query	4,096,800.108	Invalid Property Action
4,096,800.101	Invalid Query Object Handle								
4,096,800.105	Invalid Index								
4,096,800.106	Active Query								
4,096,800.108	Invalid Property Action								

Example:

```

EN N QUERY
D OPEN^SDQ( .QUERY)
>>> IF '$$ERRCHK^SDQUT()' D INDEX^SDQ( .QUERY, "PATIENT/DATE", "SET" )
      IF '$$ERRCHK^SDQUT()' D PAT^SDQ( .QUERY, 101, "SET" )
      IF '$$ERRCHK^SDQUT()' D DATE^SDQ( .QUERY, 2970501, 2970601, "SET" )
      IF '$$ERRCHK^SDQUT()' D SCANCB^SDQ( .QUERY, "D CB^XXSCAN(Y,Y0,
          .SDSTOP)", "SET" )
      IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ( .QUERY, "TRUE", "SET" )
      IF '$$ERRCHK^SDQUT()' D SCAN^SDQ( .QUERY, "FORWARD" )
      D CLOSE^SDQ( .QUERY)
      Q
      ;
CB(SDOE, SDOE0, SDSTOP)      ; -- callback logic
W !, SDOE, " >>> ", SDOE0
Q

```

ID:	86				
Name:	SDQ EOF				
Declaration:	\$\$EOF^SDQ(query [,errors])				
Description:	This function returns a boolean that indicates whether the Query Object cursor is positioned on the last encounter record in the result set.				
	This is a read-only, run-time-only property.				
Arguments:	<table border="0"> <tr> <td>query</td> <td>Encounter Query Handle</td> </tr> <tr> <td>errors</td> <td>Error Array [optional]</td> </tr> </table>	query	Encounter Query Handle	errors	Error Array [optional]
query	Encounter Query Handle				
errors	Error Array [optional]				
Return Values:	<table border="0"> <tr> <td>1</td> <td>Yes, query cursor is at the last record or no records exist for query</td> </tr> <tr> <td>0</td> <td>No, query cursor is not at last record</td> </tr> </table>	1	Yes, query cursor is at the last record or no records exist for query	0	No, query cursor is not at last record
1	Yes, query cursor is at the last record or no records exist for query				
0	No, query cursor is not at last record				
Related RPC:	Not applicable				
Error Codes:	<table border="0"> <tr> <td>4,096,800.101</td> <td>Invalid Query Object Handle</td> </tr> <tr> <td>4,096,800.102</td> <td>Inactive Query</td> </tr> </table>	4,096,800.101	Invalid Query Object Handle	4,096,800.102	Inactive Query
4,096,800.101	Invalid Query Object Handle				
4,096,800.102	Inactive Query				
Example:	<pre> EN N QUERY D OPEN^SDQ(.QUERY) IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET") IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET") IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2971001, 2971231, "SET") IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET") IF '\$\$ERRCHK^SDQUT()' D FIRST^SDQ(.QUERY) >>> IF '\$\$ERRCHK^SDQUT()' F Q:\$EOF^S DQ(.QUERY) D . D PROCESS(.QUERY) . D NEXT^SDQ(.QUERY) D CLOSE^SDQ(.QUERY) Q ; PROCESS(SDQ, SDERR) ; -- do some process N SDOE S SDOE=+\$GETENTRY^SDQ(.SDQ, \$G(SDERR)) ; W !, SDOE, " >>> ", \$\$GETOE^SDOE(.SDOE , \$G(SDERR)) Q </pre>				

ID: 87**Name:** SDQ BOF**Declaration:** \$\$BOF^SDQ(query [,errors])**Description:** This function returns a boolean that indicates whether the Query Object cursor is positioned on the first encounter record in the result set.

This is a read-only, run-time-only property.

Arguments:
query Encounter Query Handle
errors Error Array [optional]**Return Values:**
1 Yes, query cursor is at the first record or no records exist for query
0 No, query cursor is not at first record**Related RPC:** Not applicable**Error Codes:**
4,096,800.101 Invalid Query Object Handle
4,096,800.102 Inactive Query**Example:**
EN N QUERY
D OPEN^SDQ(.QUERY)
IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET")
IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2961215, 2961215.2359, , "SET")
IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
IF '\$\$ERRCHK^SDQUT()' D LAST^SDQ(.QUERY)
>>> IF '\$\$ERRCHK^SDQUT()' F Q:\$BOF^SDQ(.QUERY) D
 . D PROCESS(.QUERY)
 . D PRIOR^SDQ(.QUERY)
D CLOSE^SDQ(.QUERY)
Q
;
PROCESS(SDQ, SDERR) ; -- do some process
N SDOE
S SDOE=+\$GETENTRY^SDQ(.SDQ, \$G(SDERR)) ;
W !, SDOE, " >>> ", \$\$GETOE^SDOE(.SDOE, \$G(SDERR))
Q

ID: 88**Name:** SDQ ACTIVE STATUS**Declaration:** ACTIVE^SDQ(query, .status, action [,errors])**Description:** This method is used to set and retrieve the Active Status property of a Query Object.

Setting this property from FALSE to TRUE will cause the query to execute producing a new result set. The query cursor will be positioned on the first record in the result set.

Setting this property from TRUE to FALSE will cause the result set to be removed.

Other methods like SDQ REFRESH and SDQ NEXT will also return an error if this property is set to FALSE.

Finally, setting the status to TRUE causes validity checks to be invoked. These checks determine whether all the necessary query properties are set correctly. If they are not set, then the status will not be changed to TRUE and an Invalid Query Property error is returned in the error array parameter.

Arguments:	query	Encounter Query Handle
	status	Encounter Query Active Status
	action	Action
	errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** Not applicable

Error Codes: 4,096,800.101 Invalid Query Object Handle
 4,096,800.103 Invalid Active Status
 4,096,800.108 Invalid Property Action
 4,096,800.109 Invalid Query Property

Example: EN N QUERY
 D OPEN^SDQ(.QUERY)
 IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
 IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET")
 IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2970501, 2970601, "SET")
 IF '\$\$ERRCHK^SDQUT()' D SCANCB^SDQ(.QUERY, "D CB^XXSCAN(Y,Y0,
 .SDSTOP)", "SET")
 >>> IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
 IF '\$\$ERRCHK^SDQUT()' D SCAN^SDQ(.QUERY, "FORWARD")
 D CLOSE^SDQ(.QUERY)
 Q
 ;
 CB(SDOE, SDOE0, SDSTOP) ; -- callbac k logic
 W !, SDOE, " >>>", SDOE0
 Q

ID: 89**Name:** SDQ COUNT**Declaration:** \$\$COUNT^SDQ(query [,errors])**Description:** This function returns the number of encounter records in the Query Object's result set.

This function takes a variable amount of time to execute depending on the result set produced by the query and whether all entries have been previously accessed by the query object

This is a read-only, run-time-only property.

Arguments:

query	Encounter Query Handle
errors	Error Array [optional]

Return Values: <number> count of records in query's result set

0	no records in the query's result set
---	--------------------------------------

null	invalid query or query not active
------	-----------------------------------

Related RPC: Not applicable

Error Codes:

4,096,800.101	Invalid Query Object Handle
4,096,800.102	Inactive Query

Example:

```

EN N QUERY
D OPEN^SDQ( .QUERY)
IF '$$ERRCHK^SDQUT()' D INDEX^SDQ( .QUERY, "PATIENT/DATE", "SET" )
IF '$$ERRCHK^SDQUT()' D PAT^SDQ( .QUERY,101, "SET" )
IF '$$ERRCHK^SDQUT()' D DATE^SDQ( .QUERY,2970501,2970601, "SET" )
IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ( .QUERY, "TRUE", "SET" )
>>> IF '$$ERRCHK^SDQUT()' W !,"Count: ", $$COUNT^SDQ( .QUERY)
D CLOSE^SDQ( .QUERY)
Q

--> Count: 5

```

ID:	90				
Name:	SDQ FIRST				
Declaration:	FIRST^SDQ(query [,errors])				
Description:	This method positions the query cursor at the first encounter record in the Query Object's result set.				
Arguments:	<table border="0"> <tr> <td>query</td> <td>Encounter Query Handle</td> </tr> <tr> <td>errors</td> <td>Error Array [optional]</td> </tr> </table>	query	Encounter Query Handle	errors	Error Array [optional]
query	Encounter Query Handle				
errors	Error Array [optional]				
Return Values:	Not applicable for procedures				
Related RPC:	Not applicable				
Error Codes:	4,096,800.101 Invalid Query Object Handle 4,096,800.102 Inactive Query				

Example:

```

EN  N QUERY
D OPEN^SDQ( .QUERY)
IF '$$ERRCHK^SDQUT()' D INDEX^SDQ( .QUERY, "PATIENT/DATE", "SET" )
IF '$$ERRCHK^SDQUT()' D PAT^SDQ( .QUERY,101, "SET" )
IF '$$ERRCHK^SDQUT()' D DATE^SDQ( .QUERY,2971215,2971221, "SET" )
IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ( .QUERY, "TRUE", "SET" )
>>> IF '$$ERRCHK^SDQUT()' D FIRST^SDQ( .QUERY)
      IF '$$ERRCHK^SDQUT()' F Q:$EOF^SDQ( .QUERY) D
        . D PROCESS( .QUERY)
        . D NEXT^SDQ( .QUERY)
      D CLOSE^SDQ( .QUERY)
      Q
      ;
PROCESS(SDQ,SDERR)           ; -- do some process
N SDOE
S SDOE=+$GETENTRY^SDQ( .SDQ,$G(SDERR)) ;
W ! ,SDOE," >>>",$GETOE^SDOE($DOE,$G(SDERR))
Q

```

ID:	91				
Name:	SDQ LAST				
Declaration:	LAST^SDQ(query [,errors])				
Description:	This method positions the query cursor at the last encounter record in the Query Object's result set.				
Arguments:	<table border="0"> <tr> <td>query</td> <td>Encounter Query Handle</td> </tr> <tr> <td>errors</td> <td>Error Array [optional]</td> </tr> </table>	query	Encounter Query Handle	errors	Error Array [optional]
query	Encounter Query Handle				
errors	Error Array [optional]				
Return Values:	Not applicable for procedures				
Related RPC:	Not applicable				
Error Codes:	4,096,800.101 Invalid Query Object Handle 4,096,800.102 Inactive Query				

Example:

```

EN  N QUERY
D OPEN^SDQ( .QUERY)
IF '$$ERRCHK^SDQUT()' D INDEX^SDQ( .QUERY, "PATIENT/DATE", "SET" )
IF '$$ER RCHK^SDQUT()' D PAT^SDQ( .QUERY, 101, "SET" )
IF '$$ERRCHK^SDQUT()' D DATE^SDQ( .QUERY, 2960415, 2970415, "SET" )
IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ( .QUERY, "TRUE", "SET" )
>>> IF '$$ERRCHK^SDQUT()' D LAST^SDQ( .QUERY)
      IF '$$ERRCHK^SDQUT()' F Q:$BOF^SDQ( .QUERY) D
        . D PROCESS( .QUERY)
        . D PRIOR^SDQ( .QUERY)
      D CLOSE^SDQ( .QUERY)
      Q
      ;
PROCESS( SDQ, SDERR)      ; -- do some process
N SDOE
S SDOE=+$GETENTRY^SDQ( .SDQ, $G( SDERR)) ;
W ! , SDOE, " >>>" , $$GETOE^SDOE( .SDOE, $G( SDERR)) 
Q

```

ID:	92						
Name:	SDQ NEXT						
Declaration:	NEXT^SDQ(query [,errors])						
Description:	This method positions the query cursor at the next encounter record in the Query Object's result set.						
Arguments:	<table border="0"> <tr> <td>query</td> <td>Encounter Query Handle</td> </tr> <tr> <td>errors</td> <td>Error Array [optional]</td> </tr> </table>	query	Encounter Query Handle	errors	Error Array [optional]		
query	Encounter Query Handle						
errors	Error Array [optional]						
Return Values:	Not applicable for procedures						
Related RPC:	Not applicable						
Error Codes:	<table border="0"> <tr> <td>4,096,800.101</td> <td>Invalid Query Object Handle</td> </tr> <tr> <td>4,096,800.102</td> <td>Inactive Query</td> </tr> <tr> <td>4,096,800.111</td> <td>End of File</td> </tr> </table>	4,096,800.101	Invalid Query Object Handle	4,096,800.102	Inactive Query	4,096,800.111	End of File
4,096,800.101	Invalid Query Object Handle						
4,096,800.102	Inactive Query						
4,096,800.111	End of File						
Example:	<pre> EN N QUERY D OPEN^SDQ(.QUERY) IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET") IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY,101,"SET") IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY,2970501,2970615,"SET") IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY,"TRUE","SET") IF '\$\$ERRCHK^SDQUT()' D FIRST^SDQ(.QUERY) IF '\$\$ERRCHK^SDQUT()' F Q:\$EOF^SDQ(.QUERY) D . D PROCESS(.QUERY) >>> . D NEXT^SDQ(.QUERY) D CLOSE^SDQ(.QUERY) Q ; PROCESS(SDQ,SDERR) ; -- do some process N SDOE S SDOE=+\$GETEN TRY^SDQ(.SDQ,\$G(SDERR)) ; W !,SDOE," >>>",\$GETOE^SDOEOE(.SDOE,\$G(SDERR)) Q </pre>						

ID:	93						
Name:	SDQ PRIOR						
Declaration:	PRIOR^SDQ(query [,errors])						
Description:	This method positions the query cursor at the prior encounter record in the Query Object's result set.						
Arguments:	<table border="0"> <tr> <td>query</td> <td>Encounter Query Handle</td> </tr> <tr> <td>errors</td> <td>Error Array [optional]</td> </tr> </table>	query	Encounter Query Handle	errors	Error Array [optional]		
query	Encounter Query Handle						
errors	Error Array [optional]						
Return Values:	Not applicable for procedures						
Related RPC:	Not applicable						
Error Codes:	<table border="0"> <tr> <td>4,096,800.101</td> <td>Invalid Query Object Handle</td> </tr> <tr> <td>4,096,800.102</td> <td>Inactive Query</td> </tr> <tr> <td>4,096,800.110</td> <td>Beginning of File</td> </tr> </table>	4,096,800.101	Invalid Query Object Handle	4,096,800.102	Inactive Query	4,096,800.110	Beginning of File
4,096,800.101	Invalid Query Object Handle						
4,096,800.102	Inactive Query						
4,096,800.110	Beginning of File						

Example: EN N QUERY
D OPEN^SDQ(.QUERY)
IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET")
IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2970 101, 2971231, "SET")
IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
IF '\$\$ERRCHK^SDQUT()' D LAST^SDQ(.QUERY)
IF '\$\$ERRCHK^SDQUT()' F Q:\$BOF^SDQ(.QUERY) D
. D PROCESS(.QUERY)
>>> . D PRIOR^SDQ(.QUERY)
D CLOSE^SDQ(.QUERY)
Q
;
PROCESS(SDQ, SDERR) ; -- do some process
N SDOE
S SDOE=+\$GETENTRY^SDQ(.SDQ, \$G(SDERR)) ;
W !, SDOE, " >>> ", \$GETOE^SDOE(\$DOE, \$G(SDERR))
Q

ID:	94				
Name:	SDQ REFRESH				
Declaration:	REFRESH^SDQ(query [,errors])				
Description:	This method causes the Query Object to be re-executed and produce a new result set based on the most recent data in the database.				
	This method does the same thing as setting the Active Status property from TRUE to FALSE and then FALSE to TRUE.				
	Also, this method positions the query cursor at the first encounter record in the result set.				
Arguments:	<table border="0"> <tr> <td>query</td> <td>Encounter Query Handle</td> </tr> <tr> <td>errors</td> <td>Error Array [optional]</td> </tr> </table>	query	Encounter Query Handle	errors	Error Array [optional]
query	Encounter Query Handle				
errors	Error Array [optional]				
Return Values:	Not applicable for procedures				
Related RPC:	Not applicable				
Error Codes:	<table border="0"> <tr> <td>4,096,800.101</td> <td>Invalid Query Object Handle</td> </tr> <tr> <td>4,096,800.102</td> <td>Inactive Query</td> </tr> </table>	4,096,800.101	Invalid Query Object Handle	4,096,800.102	Inactive Query
4,096,800.101	Invalid Query Object Handle				
4,096,800.102	Inactive Query				
Example:	<pre> EN N QUERY D OPEN^SDQ(.QUERY) IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET") IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET") IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2970501, 2970601, "SET") IF '\$\$ERRCHK^SDQUT()' D SCANCB^SDQ(.QUERY, "D CB^XXSCAN(Y,Y0, .SDSTOP)", "SET") IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET") IF '\$\$ERRCHK^SDQUT()' D SCAN^SDQ(.QUERY, "FORWARD") IF '\$\$ERRCHK^SDQUT()' D CHANGE^XXEDIT >>> IF '\$\$ERRCHK^SDQUT()' D REFRESH^SDQ(.QUERY) IF '\$\$ERRCHK^SDQUT()' D SCAN^SDQ(.QUERY) D CLOSE^SDQ(.QUERY) Q ; CB(SDOE,SDOE0,SDSTOP) ; -- callback logic W !,SDOE," >>>,SDO E0 Q </pre>				

ID: 95

Name: SDQ GET CURRENT ENTRY ID

Declaration: \$\$GETENTRY^SDQ(query [,errors])

Description: This function returns the internal entry number to the OUTPATIENT ENCOUNTER file (#409.68) for the encounter record at the current Query Object cursor position.

This is a read-only property.

Arguments:

query	Encounter Query Handle
errors	Error Array [optional]

Return Values:

<pointer>	ID for entry
<null>	if no entries in result set

Related RPC: Not applicable

Error Codes:

4,096,800.101	Invalid Query Object Handle
4,096,800.102	Inactive Query

Example:

```

EN N QUERY
D OPEN^SDQ(.QUERY)
IF '$$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
IF '$$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET")
IF '$$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2961201, 2961231.2359,
    "SET")
IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
IF '$$ERRCHK^SDQUT()' D LAST^SDQ(.QUERY)
>>> W !,"Entry: ", $$GETENTRY^SDQ(.QUERY)
D CLOSE^SDQ(.QUERY)
Q

--> Entry: 3838221

```

ID: 98

Name: SDOE GET ZERO NODE

Declaration: \$\$GETOE^SDOE(encounter [,errors])

Description: This function returns the zeroth node of an outpatient encounter.

NOTE: Currently (7/97), only fields .01 thru .08 and .1 thru .13 of the zeroth are returned. Other nodes and fields are not supported.

For detail information regarding the fields, see the data dictionary for the OUTPATIENT ENCOUNTER file (#409.68).

Arguments: encounter Encounter IEN
 errors Error Array [optional]

Return Values: Zeroth node of outpatient encounter or null

NOTE: Only supported fields are returned. Those fields not supported/returned are null.

Related RPC: SDOE GET ZERO NODE

Error Codes: 4,096,800.001 Invalid Encounter ID

Example: >>> W \$\$GETOE^SDOE(4592)

```
--> 2970602. 08^706^144^62^407^^2970805.1107^1^^9^1^2^10
```

ID: 99**Name:** SDQ SCAN**Declaration:** SCAN^SDQ(query [,direction] [,errors])**Description:** This procedure scans encounter records that meet criteria defined in the Query Object. For each encounter record meeting the criteria, the Callback property is executed. (See the API definition for SDQ SCAN CALLBACK for more information.)

NOTE: This procedure tends to be faster than using the SDQ NEXT/SDQ EOF scan approach when many encounter records must be processed.

NOTE: Default scan direction is FORWARD.

Arguments:

query	Encounter Query Handle
scan_direction	Direction [optional]
errors	Error Array [optional]

Return Values: Not applicable for procedures**Related RPC:** Not applicable

Error Codes:

4,096,800.101	Invalid Query Object Handle
4,096,800.102	Inactive Query
4,096,800.112	No Scan Callback Property

Example:

```

EN N QUERY
D OPEN^SDQ(.QUERY)
IF '$$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
IF '$$ERRCHK^SDQUT()' D PAT^S DQ(.QUERY, 101, "SET")
IF '$$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2970501, 2970601, "SET")
IF '$$ERRCHK^SDQUT()' D SCANCB^SDQ(.QUERY, "D CB^XXSCAN(Y, Y0,
    .SDSTOP)", "SET")
IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
>>> IF '$$ERRCHK^SDQUT()' D SCAN^SDQ(.QUERY, "FORWARD")
D CLOSE^SDQ(.QUERY)
Q
;
CB(SDOE, SDOE0, SDSTOP)      ; -- callback logic
W !, SDOE, " >>> ", SDOE0
Q

```

ID:	100								
Name:	SDQ SCAN CALLBACK								
Declaration:	SCANCB^SDQ(query, .callback, action [,errors])								
Description:	This method is used to set and retrieve the Callback property of a Query Object.								
This property is application M code executed by the query object SDQ SCAN method for each entry found in the encounter record scan process.									
The following local variables are available at run-time:									
<table border="0"> <tr> <td>Y</td><td>current encounter entry number</td></tr> <tr> <td>Y0</td><td>zeroth node of current encounter entry (only supported fields)</td></tr> </table>		Y	current encounter entry number	Y0	zeroth node of current encounter entry (only supported fields)				
Y	current encounter entry number								
Y0	zeroth node of current encounter entry (only supported fields)								
Arguments:	<table border="0"> <tr> <td>query</td><td>Encounter Query Handle</td></tr> <tr> <td>callback</td><td>Scan Callback Logic</td></tr> <tr> <td>action</td><td>Action</td></tr> <tr> <td>errors</td><td>Error Array [optional]</td></tr> </table>	query	Encounter Query Handle	callback	Scan Callback Logic	action	Action	errors	Error Array [optional]
query	Encounter Query Handle								
callback	Scan Callback Logic								
action	Action								
errors	Error Array [optional]								
Return Values:	Not applicable for properties								
Related RPC:	Not applicable								
Error Codes:	<table border="0"> <tr> <td>4,096,800.101</td><td>Invalid Query Object Handle</td></tr> <tr> <td>4,096,800.106</td><td>Active Query</td></tr> <tr> <td>4,096,800.108</td><td>Invalid Property Action</td></tr> <tr> <td>4,096,800.113</td><td>Invalid Scan Callback</td></tr> </table>	4,096,800.101	Invalid Query Object Handle	4,096,800.106	Active Query	4,096,800.108	Invalid Property Action	4,096,800.113	Invalid Scan Callback
4,096,800.101	Invalid Query Object Handle								
4,096,800.106	Active Query								
4,096,800.108	Invalid Property Action								
4,096,800.113	Invalid Scan Callback								
Example:	<pre> EN N QUERY D OPEN^SDQ(.QUERY) IF '\$\$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET") IF '\$\$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY, 101, "SET") IF '\$\$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY, 2970501, 2970601, "SET") >>> IF '\$\$ERRCHK^SDQUT()' D SCANCB^SDQ(.QUERY, "D CB^XXSCAN(Y,Y0, .SDSTOP)", "SET") IF '\$\$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET") IF '\$\$ERRCHK^SDQUT()' D SCAN^SDQ(.QUERY, "FORWARD") D CLOSE^SDQ(.QUERY) Q ; CB(SDOE, SDOE0, SDSTOP) ; -- callback logic W !, SDOE, " >>>", SDOE0 Q </pre>								

ID:	101
Name:	SDQ ERROR CHECK
Declaration:	<code>\$\$ERRCHK^SDQUT([errors])</code>
Description:	This general utility function returns a boolean that indicates whether the current error array contains any errors.
	The current error array is the array indicated by the optional “error” parameter. If this “error” parameter is not specified, the standard <code>^TMP(“DIERR”,\$J)</code> is used.
Arguments:	errors Error Array [optional]
Return Values:	1 Yes, at least one error is in the error array 0 No, no errors are in the error array
Related RPC:	Not applicable
Error Codes:	None
Example:	<pre>>>> W \$\$ERRCHK^SDQUT() --> 1</pre>

Application Programmer Interfaces

Remote Procedure Calls

Name: SDOE ASSIGNED A DIAGNOSIS

Description: This Remote Procedure Call (RPC) returns a boolean indicator on whether at least one diagnoses has been associated with an encounter.

Related API: 64 - SDOE ASSIGNED A DIAGNOSIS

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE ASSIGNED A DIAGNOSIS';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Remote Procedure Calls

Name: SDOE ASSIGNED A PROCEDURE

Description: This Remote Procedure Call (RPC) returns a boolean indicator on whether at least one procedure has been associated with an encounter.

Related API: 65 - SDOE ASSIGNED A PROCEDURE

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE ASSIGNED A PROCEDURE';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE ASSIGNED A PROVIDER

Description: This Remote Procedure Call (RPC) returns a boolean indicator on whether at least one provider has been associated with an encounter.

Related API: 63 - SDOE ASSIGNED A PROVIDER

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE ASSIGNED A PROVIDER';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Remote Procedure Calls

Name: SDOE FIND DIAGNOSIS

Description: This Remote Procedure Call (RPC) returns a boolean indicator on whether a specific diagnosis is associated with an encounter.

Related API: 70 - SDOE FIND DIAGNOSIS

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes
1	Diagnosis IEN	Literal	Yes

Delphi Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE FIND DIAGNOSIS';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtDiagnosis.Text;
  broker.Param[1].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE FIND FIRST ENCOUNTER

Description: This Remote Procedure Call (RPC) returns the internal entry number of an OUTPATIENT ENCOUNTER file (#409.68) entry for the first encounter for a patient in a specified date range.

Related API: 74 - SDOE FIND FIRST ENCOUNTER

Parameter Position	Argument	Type	Required
0	Patient ID	Literal	Yes
1	Begin Date/Time	Literal	Yes
2	End Date/Time	Literal	Yes
3	Search Flags	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE FIND FIRST ENCOUNTER';
  broker.Param[0].Value := txtPatient.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtBegin.Text;
  broker.Param[1].PType := literal;
  broker.Param[2].Value := txtEnd.Text;
  broker.Param[2].PType := literal;
  broker.Param[3].Value := txtFlags.Text;
  broker.Param[3].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE FIND FIRST STANDALONE

Description: This Remote Procedure Call (RPC) returns the internal entry number of an OUTPATIENT ENCOUNTER file (#409.68) entry for the first standalone add/edit for a patient in a specified date range. Use same date for begin and end dates for specific (single) date check. Standalone encounter is an encounter with no parent and the originating process is "Stop Code Addition".

Related API: 72 - SDOE FIND FIRST STANDALONE

Parameter Position	Argument	Type	Required
0	Patient ID	Literal	Yes
1	Begin Date/Time	Literal	Yes
2	End Date/Time	Literal	Yes
3	Search Flags	Literal	Yes

Delphi Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE FIND FIRST STANDALONE';
  broker.Param[0].Value := txtPatient.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtBegin.Text;
  broker.Param[1].PType := literal;
  broker.Param[2].Value := txtEnd.Text;
  broker.Param[2].PType := literal;
  broker.Param[3].Value := txtFlags.Text;
  broker.Param[3].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE FIND LAST STANDALONE

Description: This Remote Procedure Call (RPC) returns the internal entry number of an OUTPATIENT ENCOUNTER file (#409.68) entry for the last standalone add/edit for a patient in a specified date range. Standalone encounter is an encounter with no parent and the originating process is "Stop Code Addition".

Related API: 75 - SDOE FIND LAST STANDALONE

Parameter Position	Argument	Type	Required
0	Patient ID	Literal	Yes
1	Begin Date/Time	Literal	Yes
2	Search Flags	Literal	Yes

Delphi Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE FIND LAST STANDALONE';
  broker.Param[0].Value := txtPatient.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtBegin.Text;
  broker.Param[1].PType := literal;
  broker.Param[2].Value := txtFlags.Text;
  broker.Param[2].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Remote Procedure Calls

Name: SDOE FIND PROCEDURE

Description: This Remote Procedure Call (RPC) returns a boolean indicator on whether a specific procedure is associated with an encounter.

Related API: 71 - SDOE FIND PROCEDURE

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes
1	CPT IEN	Literal	Yes

Delphi Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE FIND PROCEDURE';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtProcedure.Text;
  broker.Param[1].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE FIND PROVIDER

Description: This Remote Procedure Call (RPC) returns a boolean indicator on whether a specific provider is associated with an encounter.

Related API: 69 - SDOE FIND PROVIDER

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes
1	Practitioner ID	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE FIND PROVIDER';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtProvider.Text;
  broker.Param[1].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE GET DIAGNOSES

Description: This Remote Procedure Call (RPC) returns an array of diagnoses for an encounter.

NOTE: For encounters before 10/1/96, only scheduling data in the OUTPATIENT DIAGNOSIS file (#409.43) may exist. It will only exist if the site required diagnoses as part of the check out process. This RPC will attempt to find this "old" data, reformat the data to meet the V POV structure and return the list of diagnoses as described in API# 56. (Only the diagnosis code internal entry number is available for "old" encounters.)

Related API: 56 - SDOE GET DIAGNOSES

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE GET DIAGNOSES';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE GET GENERAL DATA

Description: This Remote Procedure Call (RPC) returns the zeroth and other nodes of an outpatient encounter entry.

Format: <encounter node>;<node value>

NOTE: Currently (7/97) only the zeroth node is returned. Also, only fields .01 thru .08 and .1 thru .13 of the zeroth node are returned. Other nodes and fields are not supported.

Related API: 76 - SDOE GET GENERAL DATA

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE GET GENERAL DATA';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE GET PRIMARY DIAGNOSIS

Description: This Remote Procedure Call (RPC) returns the internal entry number of the primary diagnosis code (^ICD9) for an encounter.

NOTE: For encounters before 10/1/96, this RPC will always return 0. This primary diagnosis was not retrieved nor stored by the system for these "old" encounters.

Related API: 73 - SDOE GET PRIMARY DIAGNOSIS

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE GET PRIMARY DIAGNOSIS';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE GET PROCEDURES

Description: This Remote Procedure Call (RPC) returns a subscripted array of CPTs for an encounter.

NOTE: For encounters before 10/1/96, only scheduling data in the SCHEDULING VISITS file (#409.5) may exist. It will only exist if the site required procedures as part of the check out process. This RPC will attempt to find this "old" data, reformat the data to meet the V CPT structure and return the list of procedures as described in API# 61. (Only the CPT code internal entry number and count are available for "old" encounters.)

Related API: 61 - SDOE GET PROCEDURES

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE GET PROCEDURES';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE GET PROVIDERS**Description:** This Remote Procedure Call (RPC) returns a subscripted array of providers for an encounter.

NOTE: For encounters before 10/1/96, only scheduling data in the OUTPATIENT PROVIDER file (#409.44) may exist. It will only exist if the site required provider as part of the check out process. This RPC will attempt to find this "old" data, reformat the data to meet the V PROVIDER structure and return the list of providers as described in API# 58. (Only the provider internal entry number is available for "old" encounters.)

Related API: 58 - SDOE GET PROVIDERS

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi Example:

```

begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE GET PROVIDERS';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;

```

Name: SDOE GET ZERO NODE

Description: This Remote Procedure Call (RPC) returns the zeroth node of an outpatient encounter.

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the zeroth node are returned.

Related API: 98 - SDOE GET ZERO NODE

Parameter Position	Argument	Type	Required
0	Encounter IEN	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE GET ZERO NODE';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE LIST ENCOUNTERS FOR DATES

Description: This Remote Procedure Call (RPC) returns a list of outpatient encounters for a date range.

Format: <encounter ien>;<zeroth node of encounter>

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the zeroth node are returned.

Related API: 99 - SDQ SCAN

Parameter Position	Argument	Type	Required
0	Begin Date/Time	Literal	Yes
1	End Date/Time	Literal	Yes

Delphi

Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE LIST ENCOUNTERS FOR DATES';
  broker.Param[0].Value := txtBegin.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtEnd.Text;
  broker.Param[1].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE LIST ENCOUNTERS FOR PAT

Description: This Remote Procedure Call (RPC) returns a list of outpatient encounters for a specified patient and specified date range.

Format: <encounter ien>;<zeroth node of encounter>

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the zeroth node are returned.

Related API: 99 - SDQ SCAN

Parameter Position	Argument	Type	Required
0	Patient ID	Literal	Yes
1	Begin Date/Time	Literal	Yes
2	End Date/Time	Literal	Yes

Delphi Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE LIST ENCOUNTERS FOR PAT';
  broker.Param[0].Value := txtPatient.Text;
  broker.Param[0].PType := literal;
  broker.Param[1].Value := txtBegin.Text;
  broker.Param[1].PType := literal;
  broker.Param[2].Value := txtEnd.Text;
  broker.Param[2].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE LIST ENCOUNTERS FOR VISIT

Description: This Remote Procedure Call (RPC) returns a list of outpatient encounters for a specified visit.

Format: <encounter ien>;<zeroth node of encounter>

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the zeroth node are returned.

Related API: 99 - SDQ SCAN

Parameter Position	Argument	Type	Required
0	Visit IEN	Literal	Yes

Delphi Example:

```
begin
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE LIST ENCOUNTERS FOR VISIT';
  broker.Param[0].Value := txtVisit.Text;
  broker.Param[0].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Name: SDOE PARSE GENERAL DATA

Description: This Remote Procedure Call (RPC) will parse the data returned by the "SDOE GET GENERAL DATA" RPC into individual field nodes.

Format: <encounter field #>; <field value>

Related API: 78 - SDOE PARSE GENERAL DATA

Parameter Position	Argument	Type	Required
0	Encounter Data	List	Yes
1	Encounter Parse Format	Literal	Yes

Delphi Example:

```
begin
  // get data needed for the 'parse' RPC
  memResults.Lines.Clear;
  broker.RemoteProcedure := 'SDOE GET GENERAL DATA';
  broker.Param[0].Value := txtEncounter.Text;
  broker.Param[0].PType := literal;
  broker.Call;

  // call parser RPC using data retrieved from above
  broker.RemoteProcedure := 'SDOE PARSE GENERAL DATA';
  broker.Param[0].Value := '.SDY';
  broker.Param[0].PType := list;
  broker.Param[0].Mult['0'] := Piece(broker.Results[0],':',2);
  broker.Param[1].Value := txtFormat.Text;
  broker.Param[1].PType := literal;
  broker.Call;
  memResults.Lines := broker.Results;
end;
```

Remote Procedure Calls

Error Processing

Structure

Most of the APIs defined in the ACRP Interface Toolkit have error messages that may be passed back to the calling application in the event that an error does occur. The structure and design of these error messages is the same as that used by the VA FileMan Database Server APIs.

The structure is based on the DIALOG file (# .84) and supporting utilities. For detailed discussion on the structure and utilities, see the following in the VA FileMan Programmer Manual:

Section	Sub-Section
Database Server	How the FileMan Database Server Communicates
Database Server	MSG^DIALOG() This is the utility API that a developer can use to help process returning error messages.
Database Server	BLD^DIALOG()
Database Server	\$\$EZBLD^DIALOG()

Error Arrays

As described in the VA FileMan documentation, error messages are stored, by default, in the ^TMP("DIERR",\$J) array. However, a developer can explicitly indicate the name of an array where all error messages should be placed. This array name is passed as a parameter as the following example shows:

```
S X=$$GETOE^SDOE(OE,"MYERROR")
```

If any errors occurred during this function call, the errors will be placed in the MYERROR array.

If the following call was made instead...

```
S X=$$GETOE^SDOE(OE)
```

...then any errors would be placed as nodes in ^TMP("DIERR",\$J) ARRAY.

Error Processing

Whenever the developer makes an AIT call and does not explicitly indicate an error message array, any nodes in the ^TMP("DIERR",\$J) array are killed. As a result, the developer will know that any data that exists after the call was generated by that call. If an array is explicitly specified, then it is the responsibility of the developer to manage that array as required by the application.

Finally, in order to make sure that the arrays are deleted before the application quits, use the following call that deletes the arrays and related variables:

```
DO CLEAN^DILF
```

If the developer has instructed the AIT call to place error messages into a specific array, the developer must be sure to clean up that array.

For documentation on CLEAN^DILF, see the following in the VA FileMan Programmer Manual:

Section	Sub-Section
Database Server	CLEAN^DILF

AIT ERROR PROCESSING TOOLS

Checking For Errors

Many times when making AIT API calls, the developer will have already determined that parameters being passed are valid, such as passing in a date range or a patient internal entry number. As a result, there is no need to look for errors.

However, in the event there is a possibility that an error may be encountered by the AIT API, the developer can use the \$\$ERRCHK^SDQUT call to easily determine if an error has occurred, as the following example shows:

```

EN  N QUERY
D OPEN^SDQ(.QUERY)
IF '$$ERRCHK^SDQUT()' D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
IF '$$ERRCHK^SDQUT()' D PAT^SDQ(.QUERY,101,"SET")
IF '$$ERRCHK^SDQUT()' D DATE^SDQ(.QUERY,2970101,2971231,"SET")
IF '$$ERRCHK^SDQUT()' D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
IF '$$ERRCHK^SDQUT()' D LAST^SDQ(.QUERY)
IF '$$ERRCHK^SDQUT()' F Q:$BOF^SDQ(.QUERY)  D
.  D PROCESS(.QUERY)
.  D PRIOR^SDQ(.QUERY)
D CLOSE^SDQ(.QUERY)
D CLEAN^DILF
Q

```

For more information on \$\$ERRCHK^SDQUT, see its API definition in this manual.

Debugging

Another tool available to the developer and Customer Service staff is the SDEBUG variable. If this variable is defined, the AIT will automatically display any errors.

For example, if there is an error in the date range specified for a call to DATE^SDQ(), then the error will be displayed, as the following shows:

Code:

```

EN  N QUERY
S SDEBUG=""
D OPEN^SDQ(.QUERY)
D INDEX^SDQ(.QUERY, "PATIENT/DATE", "SET")
D PAT^SDQ(.QUERY,101,"SET")
D DATE^SDQ(.QUERY, 2980101, 2971201, "SET")
D ACTIVE^SDQ(.QUERY, "TRUE", "SET")
D LAST^SDQ(.QUERY)
F Q:$BOF^SDQ(.QUERY)  D
.  D PROCESS(.QUERY)
.  D PRIOR^SDQ(.QUERY)
D CLOSE^SDQ(.QUERY)
D CLEAN^DILF
K SDEBUG
Q

```

Error Display: Error Number: 4096800.022

Date range is not valid.

Date Range: '2980101' to '2971201'.

Error Processing

As shown in the example, the developer can temporarily set the SDEBUG variable directly in the routine during testing. Customer Service personnel could do the same or as part of an ENTRY ACTION for an Option or Protocol.

Appendix - Argument Definitions

Name:	Action
Description:	This parameter indicates whether a property should be set or retrieved.
	<u>Valid Values</u>
	GET retrieve property
	SET set property
Data Type:	String
Formal List Variable:	action

Name:	Begin Date/Time
Description:	Beginning Date and time.
	Format: VA FileMan
	Time: Optional
Data Type:	Date/Time
Formal List Variable:	begin_date

Name:	CPT IEN
Description:	This is the internal entry number of an entry in the CPT [#81 - ^ICPT] file.
Data Type:	Pointer
Formal List Variable:	cpt

Appendix - Argument Definitions

Name:	Diagnosis IEN
Description:	This is the internal entry number of an entry in the ICD DIAGNOSIS [#80 - ^ICD9] file.
Data Type:	Pointer
Formal List Variable:	diagnosis
Name:	Direction [optional]
Description:	This parameter indicates whether a search should start at the beginning of the list and work down or start at the end of the list and work up.
	<u>Valid Values</u>
	FORWARD start at beginning and work down
	BACKWARD start at end and work up
Data Type:	String
Formal List Variable:	scan_direction
Name:	Encounter Data
Description:	This array contains subscripts that correspond to each node of data for an outpatient encounter entry.
	NOTE: Currently (7/97) only the zeroth node is returned. Also, only fields .01 thru .08 and .1 thru .13 of the zeroth are returned. Other nodes and fields are not supported.
	For detail information regarding the fields, see the data dictionary for the OUTPATIENT ENCOUNTER file (#409.68).
Data Type:	Array Reference
Formal List Variable:	encounter_data

Name: **Encounter IEN**

Description: This is the internal entry number of an entry in the OUTPATIENT ENCOUNTER [#409.68 - ^SCE] file.

Data Type: Pointer

Formal List Variable: encounter

Name: **Encounter Parse Format**

Description: Defines format for parsed data.

Valid Values

INTERNAL	use internal format
EXTERNAL	external/display format

Data Type: String

Formal List Variable: format

Name:	Encounter Parsed Data
Description:	This parameter will return parsed encounter data. The format will be based on the Encounter Parse Format parameter, internal or external.
	Each field will have its data returned in the array using the field number as the subscript, i.e., SDY(.01)=2970712.13.
	NOTE: Currently (7/97) only the zeroth node is returned. Also, only fields .01 thru .08 and .1 thru .13 of the zeroth are returned. Other nodes and fields are not supported.
	For detail information regarding the fields, see the data dictionary for the OUTPATIENT ENCOUNTER file (#409.68).
Data Type:	Array Reference
Formal List Variable:	parsed_data
Name:	Encounter Query Active Status
Description:	This parameter indicates the status of the query object.
	<u>Valid Values</u>
	1 query active
	0 query not active
Data Type:	Boolean
Formal List Variable:	status

Name:**Encounter Query Filter****Description:**

This parameter can be used by the developer to screen out entries using M code. The code must set \$T.

The following information is available to the developer when this logic is executed:

<u>Variable</u>	<u>Description</u>
Y	Internal Encounter Entry Number
Y0	Zeroth node of encounter entry

NOTE: Depending on the file, not all fields of the zeroth node may be returned. Only those fields supported by the application will be returned. Non-supported fields will be null.

NOTE: This filter logic should only return a boolean value via \$TEST. Developers should not set application variables as part of this filter logic.

Data Type: String

Formal List Variable: filter

Name:**Encounter Query Handle****Description:**

The query handle obtained as a result of a call to the SDQ OPEN method of the query object.

More than one query object instance can be created by a process. This handle indicates which instance to use when various query object methods are called.

Data Type: String

Formal List Variable: query

Appendix - Argument Definitions

Name:	Encounter Query Index
Description:	This parameter is used to indicate which cross reference the query should use when producing the result set.
<u>Valid Values</u>	
PATIENT	use patient cross reference
PATIENT/DATE	use patient by encounter date cross reference
DATE/TIME	use the encounter date cross reference
VISIT	use the visit cross reference
Data Type:	String
Formal List Variable:	index

Name:	End Date/Time
Description:	Ending date and time.
Format:	VA FileMan
Time:	Optional
Data Type:	Date/Time
Formal List Variable:	end_date

Name:	Error Array [optional]
Description:	Literal identifying the array where error information should be stored. The calling application can then process any errors when control is returned.
	The error array is in DIALOG utility format.
Data Type:	Array Reference
Formal List Variable:	errors

Name:	List of V CPT Entries
Description:	An array of V CPT (#9000010.18) entries for a visit. The array is subscripted by the V CPT internal entry numbers.
	Each subscript entry value equals the zeroth node of the V CPT entry. See the data dictionary for more information on the value of each piece.
	The top level, non-subscripted value is a count of the number of entries found.
Data Type:	Array Reference
Formal List Variable:	cpt_list

Appendix - Argument Definitions

Name:	List of V POV Entries
Description:	An array of V POV (#9000010.07) entries for a visit. The array is subscripted by the V POV internal entry numbers.
	Each subscript entry value equals the zeroth node of the V POV entry. See the data dictionary for more information on the value of each piece.
	The top level, non-subscripted value is a count of the number of entries found.
Data Type:	Array Reference
Formal List Variable:	<code>dx_list</code>

Name:	List of V PROVIDER Entries
Description:	An array of V PROVIDER (#9000010.06) entries for a visit. The array is subscripted by the V PROVIDER internal entry numbers.
	Each subscript entry value equals the zeroth node of the V PROVIDER entry. See the data dictionary for more information on the value of each piece.
	The top level, non-subscripted value is a count of the number of entries found.
Data Type:	Array Reference
Formal List Variable:	<code>provider_list</code>

Name:	Patient ID
Description:	This is the internal entry number of an entry in the PATIENT [#2 - ^DPT] file.
Data Type:	Pointer
Formal List Variable:	<code>dfn</code>

Name:	Practitioner ID
Description:	This is the internal entry number of an entry in the NEW PERSON [#200 - ^VA(200)] file for a practitioner.
Data Type:	Pointer
Formal List Variable:	provider

Name:	Scan Callback Logic
Description:	Application M code executed by the query object SDQ SCAN method for each entry found in the record scan process.

The following variables are available at the time this callback code is executed:

<u>Variable</u>	<u>Description</u>
Y	Internal Encounter Entry Number
Y0	Zeroth node of encounter entry
SDSTOP	Set to 1 to tell Scan to stop processing

The callback code should look similar to either of the following:

>> Parameter Passing Approach:
D CALLBACK^XX(Y,Y0,.SDSTOP)

NOTE: If this approach is used, then SDSTOP must be passed by reference.

>> Classic M Approach
D CALLBACK^XX

Data Type:	String
Formal List Variable:	callback

Appendix - Argument Definitions

Name:	Search Flags				
Description:	This parameter allows developers to set specific flags that are used as an API searches encounter records. The flags indicate how the API should function.				
	<table><thead><tr><th><u>Character</u></th><th><u>Description</u></th></tr></thead><tbody><tr><td>C</td><td>Use only completed encounters</td></tr></tbody></table>	<u>Character</u>	<u>Description</u>	C	Use only completed encounters
<u>Character</u>	<u>Description</u>				
C	Use only completed encounters				
Data Type:	String				
Formal List Variable:	flags				

Name:	Visit IEN
Description:	This is the internal entry number of an entry in the VISIT [#9000010 - ^AUPNVSIT] file.
Data Type:	String
Formal List Variable:	visit